Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

NetOnZeroDXC: A package for the identification of networks out of multivariate time series via zero-delay cross-correlation

Alessio Perinelli [a], Leonardo Ricci [a,b,*]

[a] Department of Physics, University of Trento, 38123 Trento, Italy
[b] CIMeC, Center for Mind/Brain Sciences, University of Trento, 38068 Rovereto, Italy

A R T I C L E I N F O

Article history:
Received 10 April 2019
Received in revised form 8 July 2019
Accepted 29 August 2019

Keywords:
Cross correlation
Networks
Multivariate time series analysis

A B S T R A C T

The identification of networks is an issue that is relevant in many research fields. Assessing links between nodes of a network by analyzing time series associated to those nodes is a crucial topic in fields as diverse as neuroscience and climate research. In this work we present NetOnZeroDXC, a package that implements a recently published method for the assessment of networks out of multivariate time series. The method relies on the computation of zero-delay cross-correlation between pairs of time series. The software implements the various stages of the algorithm in a general way, thus making up a novel tool that can be used for network analysis in different fields.

Required Metadata

C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	Github repository
C3	Code Ocean compute capsule	
C4	Legal Code License	GNU GPL v3
C5	Code versioning system used	None
C6	Software code languages, tools, and services used	C++, GNU Scientific Libraries, wxWidgets
C7	Compilation requirements, operating environments & dependencies	Windows 64 bit. Linux (dependencies: GNU Scientific Libraries, wxWidgets library)
C8	If available Link to developer documentation/manual	User manual
C9	Support email for questions	leonardo.ricci@unitn.it

1. Introduction

The analysis of complex systems in terms of networks is a common issue of many research fields, from neuroscience [1] to climate research [2,3] and economics [4]. A network structure can be identified by first assessing links between possible network elements, henceforth referred to as *nodes*. The existence of a link is often determined by analyzing time series of an observable quantity recorded for each one of the two node candidates. In a recently published work [5], a new method was proposed to identify networks out of a set of time series. The method relies on the computation of zero-delay cross-correlation between pairs of time series: the significance of the correlation is assessed by means of surrogate data generation. The key feature of the algorithm is that it not only provides the network structure, in terms of links between nodes, but it also estimates the time scale at which links emerge, thus allowing to study the dynamic behavior of networks. The method, originally developed in the context of neuroscience, is applicable to any experimental situation in which nodes are characterized by scalar time series.

The present software package implements all the different stages of the aforementioned method, eventually producing a connectivity matrix out of a set of input time series. The package provides a graphical user interface that allows to preprocess the dataset and to trim the parameters for the analysis in a straightforward way. Moreover, the package provides a graphical interface that allows for the presentation of the results in terms of color-scale connectivity matrix.

Compared to existing tools for computing cross-correlations, our package implements a full analysis pipeline that includes the generation of surrogate time series for the estimation of the

* Corresponding author at: Department of Physics, University of Trento, 38123 Trento, Italy.
E-mail address: leonardo.ricci@unitn.it (L. Ricci).

https://doi.org/10.1016/j.softx.2019.100316

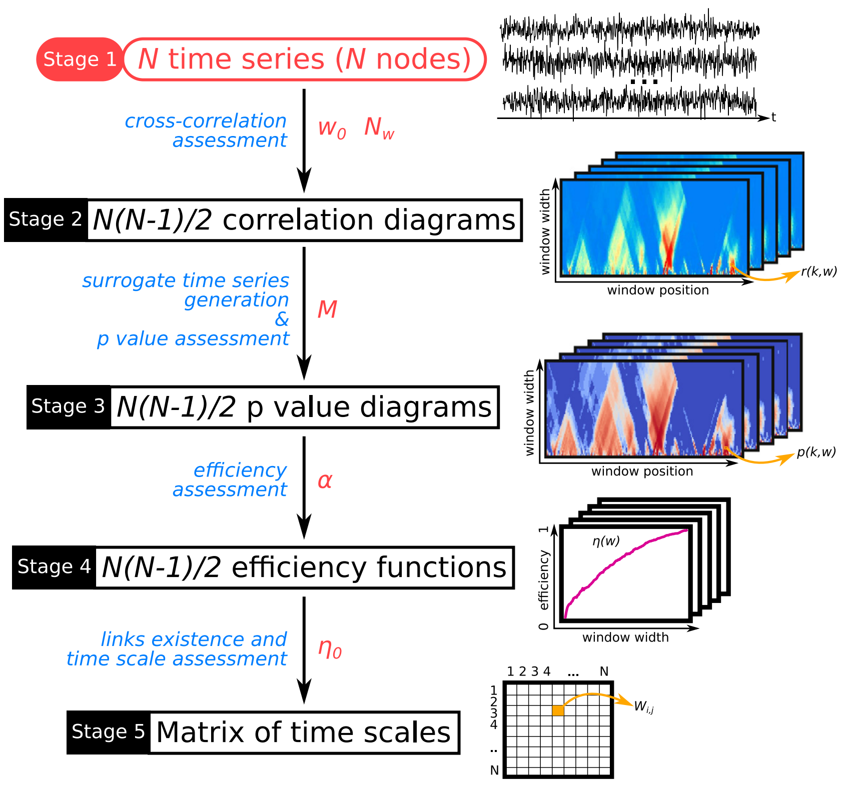


Fig. 1. Pipeline of the analysis method between stage 1, set of input time series, to stage 5, final connectivity matrix. Quantities in red correspond to input data or parameters that have to be provided by the user. The algorithm steps are highlighted in blue. All steps are described in detail in the main text. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

significance. This approach allows to tackle situations in which a non-Gaussian dynamics is present.

2. Zero-delay cross correlation analysis of networks

The method implemented in the present package is graphically summarized in Fig. 1. The input of the algorithm are N time series corresponding to the N candidate nodes (stage 1). The method automatically performs mean subtraction and normalization on the analyzed time series; no additional preprocessing is required. However, in order to suitably choose the range of window widths covered by the algorithm, a spectral analysis should be performed by means of conventional time series analysis techniques [6] and nonlinear techniques [7,8].

Prior to the main processing pipeline explained below and in order to possibly highlight underlying stationary processes, the cross-correlation of the whole time series, or alternatively the related p-value, can be computed and visualized.

The first step consists in computing a *correlation diagram* for each of the $N(N-1)/2$ pairs of nodes. Correlation is computed as the zero-delay Pearson correlation coefficient $r(k, w)$ between two *windows* of the two time series under analysis.

The width w (in terms of number of samples) of a window runs from a base width w_0 up to $N_w w_0$, where both w_0 and N_w are integer parameters set by the user; in addition, w_0 has to be even. The maximum window width must be smaller than the length of the time series Λ. The first window having maximum length $N_w w_0$ spans the points of the time series having an index between 0 and $N_w w_0 - 1$. The index of the leftmost of the two middle points (henceforth referred to as simply the window's *middle point*), namely $N_w w_0/2 - 1$, makes up the starting value k_{min} of the parameter k. This starting value remains the same also for shorter window lengths. The parameter k, which thus corresponds to the position of the middle point of any running window, is then progressively incremented in steps of w_0 up to the value that still accommodates a largest-window within the time series:

$$k = k_{min} + m \cdot w_0 \text{, where}$$
$$0 \leqslant m \leqslant m_{max} \equiv \left\lfloor \frac{\Lambda}{N_w w_0} \right\rfloor . \quad (1)$$

Consequently, a correlation diagram, i.e. the set of all correlation coefficients $r(k, w)$ for a specific pair of nodes, has N_w rows, and a number of columns $m_{max} + 1$ that depends on the length of the time series. An example of a correlation diagram is visible on the right part of Fig. 1, stage 2. A diagram of the windowing scheme for the computation of correlation coefficient is shown in Fig. 2.

In some research fields, the evaluation of the zero-delay cross-correlation can be severely spoiled by noise. In these cases, it is more convenient to evaluate $r(k, w)$ (with zero-delay between the two time series) as the average of $r(k, w; \tau)$ and $r(k, w; -\tau)$, where $r(k, w; \tau)$ ($r(k, w; -\tau)$) is the cross correlation between the first (second) time series delayed by τ and the second (first) time series. The delay τ, which has to be assigned as a positive integer number, and is therefore expressed in sampling time units, must be much smaller than the time scale to investigate. An example is provided by MEG analysis, in which the *source leakage* phenomenon spoils direct evaluations of $r(k, w)$ [9]. It has to be noted that the use of a delay changes the value of the time series length to be used in the m_{max} definition above from Λ to $\Lambda - \tau$.

The second step consists in assessing the significance of each correlation coefficient and producing a p-value diagram (stage 3), namely a set of p-values $p(k, w)$ each associated with the corresponding correlation coefficient $r(k, w)$. Given k, w, to evaluate a p-value, a set of M pairs of surrogate time series are generated out of the two original time series by means of the Iterative Amplitude Adjusted Fourier Transform (IAAFT) algorithm [10,11].

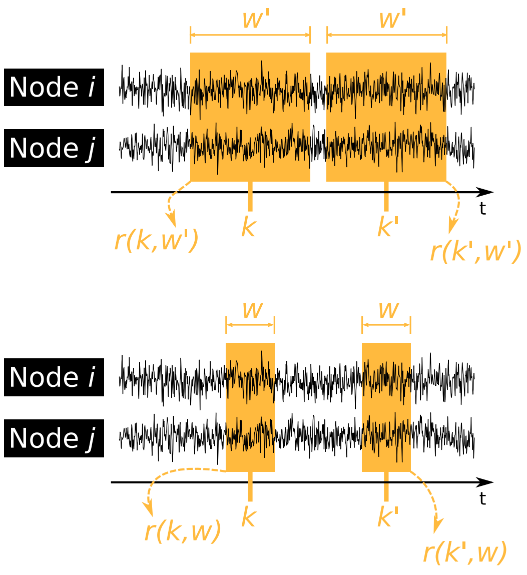


Fig. 2. Assessment of correlation coefficients $r(k, w)$ in the case of the node pair i, j.

This algorithm conserves both the distribution of values and (approximately) the autocorrelation of the original time series. A detailed discussion on limitations and caveats concerning the IAAFT algorithm can be found in Refs. [11,12]. For each pair of surrogate time series, a correlation diagram is generated. The p-value $p(k, w)$ is then computed by ranking the original $r(k, w)$ correlation coefficient within the set of the M surrogate values, and normalizing the rank by M. The number M of surrogate pairs to generate is set according to the desired resolution on the p assessment.

The significance of the cross-correlation can be also assessed by using a Fisher F-test based on a linear regression between the two time series according to the following expression:

$$p = 1 - F(f, \nu_1 = 1, \nu_2 = w - 2),$$

where $F(x, \nu_1, \nu_2)$ is the cumulative F distribution with ν_1, ν_2 degrees of freedom and

$$f = \frac{w}{\frac{1}{r^2(k,w)} - 1} .$$

However, it has to be stressed that the application of the Fisher F-test sets normality requirements that are not necessarily complied with by the time series to analyze.

The information contained in a p-value diagram is further processed by computing the *efficiency* $\eta(w)$. At a given window width w, the efficiency $\eta(w)$ is evaluated as the fraction of windows of width w for which $p(k, w) < \alpha$, where the significance threshold α is a parameter to be set by the user. This step produces an efficiency function $\eta(w)$ for each pair of nodes i, j (stage 4). Efficiency can also be computed by using non-overlapping windows, an operation that however increases the statistical noise on the parameter η.

Finally, in the last pipeline step, the existence of links and the corresponding time scale of observability are assessed. A link between node i and node j is deemed to exist if the efficiency overcomes a given efficiency threshold η_0. The minimum window width, if any, for which $\eta(W) \geqslant \eta_0$ determines the time scale of observability for the link between node i and node j. The information on the existence of links and the corresponding time scales of observability, are collected in the symmetric matrix $\widehat{W}$ (stage 5): the element $\widehat{W}_{i,j}$ of the matrix is the time scale, if any, at which a link is observable between nodes i and j. The diagonal elements of $\widehat{W}$ are equal to zero, because each node is correlated with itself at any time scale.

The matrix of time scales $\widehat{W}$ is the final product of the first part of the algorithm. However, more than one set of recordings of the same system might be available, corresponding to more than one time series for each node. For example, if $\widehat{W}_{1,A}$ is the matrix built on a first electroencephalographic recording carried out on subject A by means of a multi-channel device, one might replicate the recording in later times, thus producing $\widehat{W}_{2,A}$, $\widehat{W}_{3,A}$, ... Moreover, more than one system with the same expected network structure might be under investigation. As an example, the same EEG analysis could be carried out also on subject B, C, ... It is then necessary first to merge the information contained in the matrices of time scale within the same system and then, possibly, to merge results between different systems. These two post-processing steps are summarized in Fig. 3. Within the same system, the information contained in the matrices of time scales is merged to produce a matrix $\widehat{W}_{A,\text{merge}}$ whose i, jth element is assessed by ranking all the i, jth elements of the single matrices and picking the R_{within}th one, where R_{within} is a parameter to be set by the user. Matrix elements for which no time scale exists are considered to be larger than any finite element.

If different systems are given, each one with its $\widehat{W}_{S,\text{merge}}$, information is merged in a single matrix by applying the same ranking procedure, though by means of a different ranking parameter R_{between}. Merging information through ranking is justified by the fact that the distribution of time scales is unknown, and thus the sample mean is not necessarily a reliable statistics.

3. Package structure

The package consists of two graphical user interface (GUI) apps and two command line executables. One of the apps, `netOnZeroDXC_analysis`, performs the whole analysis pipeline from the input time series down to a matrix of time scales, as summarized in Fig. 1. The first two analysis steps, namely the computation of correlation diagrams and p-value diagrams, are also carried out by the command line program `netOnZeroDXC_diagram`. The third step, namely the computation of efficiencies, is carried out by the command line program `netOnZeroDXC_efficiency`. The two command line programs are provided separately from the GUI to facilitate a user to perform part of the analysis on remote computing clusters. Finally, the second GUI app, `netOnZeroDXC_merge`,

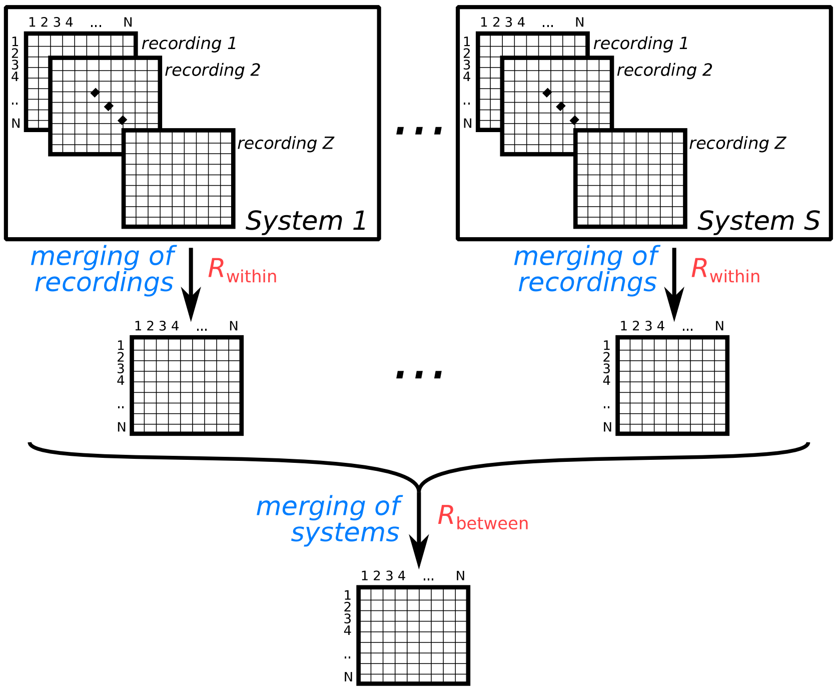


Fig. 3. Merging of the results from different recordings and systems. The two parameters R_{within} and R_{merge} have to be provided by the user.

implements the steps of the pipeline shown in Fig. 3, namely the merging of the results from different recordings within the same system and between different systems.

The code is written in C++. Besides standard C++ libraries, the GNU Scientific Library (GSL) [13,14] is employed to provide random number generation and fast Fourier transform routines. The graphical interfaces are built on the wxWidgets library [15]. The source file `netOnZeroDXC_algorithm.cpp` contains the core implementation of the algorithm, which is shared by the four programs described above. The rest of the code is devoted to input/output functionalities (`*_io.cpp`) and graphic interfaces (`*_gui_*.cpp`). The "main" source files are labeled by `*_main.cpp`. A scheme of the package components and their role in the analysis pipeline is shown in Fig. 4.

A user manual with details on the software functionalities is provided, as well as setup instructions for both Windows and Linux platforms. A set of examples is included in the package.

3.1. Generation of a matrix of time scales

All programs are fed with input data in the form of plain ASCII files with columns delimited by a valid separator (either tabulation, space or comma) to be assigned within the apps. Similarly, all results are saved as ASCII files with the same column delimiters. No constraints are set on the extension of input files. Output files are generated with a ".dat" extension.

The analysis is started by loading multivariate time series in the `netOnZeroDXC_analysis` app (or equivalently by feeding them into the command line program `netOnZeroDXC_diagram`) through a dedicated input panel. It is also possible to load intermediate analysis results instead of raw multivariate time series. In this way, it is possible to repeat some analysis steps and, for example, trim relevant parameters, while avoiding the necessity to restart the analysis procedure from the very beginning. For the same reason, all intermediate steps can be optionally saved, and sessions can be restarted from these saved steps.

When a set of files, each containing a single time series, is fed to the app, each filename has to contain a shared part and a single, running label which is sorted out by the app by looking at delimiters (either underscore or dash) and discarding shared parts (e.g. file extensions). Examples of valid file names are "sequence_a.csv", "sequence_b.csv", etc. On the other hand, in the case of a single file containing all time series, nodes are automatically labeled by progressive integer numbers. Similarly, if a set of files, each containing a single p-value diagram or a single efficiency function, is fed to the app, the corresponding filenames are required to contain a shared part and two running labels. Again, these labels are sorted out by the app in a similar way as above. Examples of valid filenames are "pdiag_12_15.dat", "pdiag_12_16.dat", etc. Running labels are lexicographically ordered.

Once data are loaded, a configuration panel allows the user to set the target stage of the analysis as well as all the required parameters (labeled in red in Fig. 1). The "Run" button triggers the computation. Further details on the functionalities of the `netOnZeroDXC_analysis` app are given in the user manual accompanying the package.

One main option concerns the possibility of exploiting, if available, parallel computing to accelerate the assessment of p-value diagrams, which is the computationally most demanding step of the pipeline. To this purpose, the command line program `netOnZeroDXC_diagram` is provided, which performs the computation of a single p-value diagram out of a pair of time series. The parameters required for this step of the analysis have to be provided to the program through command line flags. Finally, the command line program `netOnZeroDXC_efficiency` computes an efficiency function out of a single p-value diagram. For both these command line programs, the necessary flags are described in the user manual as well as in the software versions that can be invoked via -h, –help.

The `netOnZeroDXC_analysis` app can graphically show a preview of the matrix of time scales by using a color scale. Inside the preview window, two controls allow to change the significance threshold α and the efficiency threshold η_0. The preview matrix is updated accordingly in real time.

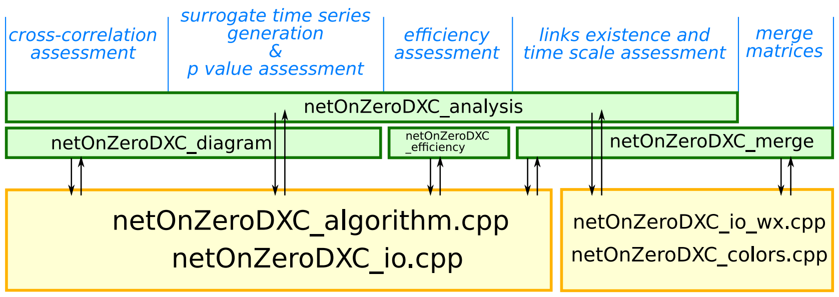
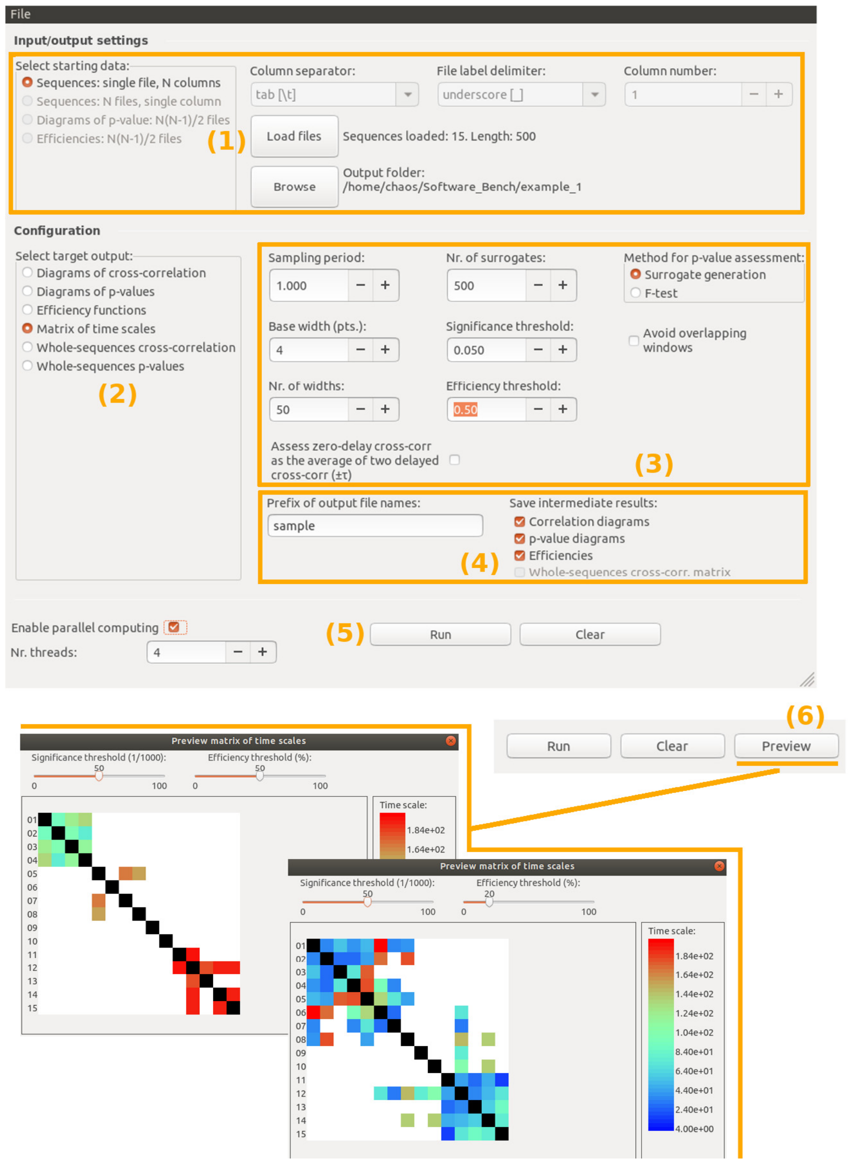


Fig. 4. Package components and their role in the analysis pipeline.



Input/output settings

Select starting data:
- [✓ Sequences: single file, N columns]
- Sequences: N files, single column
- Diagrams of p-value: N(N-1)/2 files
- Efficiencies: N(N-1)/2 files

Column separator: tab [\t]
File label delimiter: underscore [_]
Column number: 1

Sequences loaded: 15. Length: 500

Output folder: /home/chaos/Software_Bench/example_1

Configuration

Select target output:
- Diagrams of cross-correlation
- Diagrams of p-values
- Efficiency functions
- [✓ Matrix of time scales]
- Whole-sequences cross-correlation
- Whole-sequences p-values

Sampling period: 1.000
Nr. of surrogates: 500
Method for p-value assessment: [✓ Surrogate generation] F-test
Base width (pts.): 4
Significance threshold: 0.050
Avoid overlapping windows
Nr. of widths: 50
Efficiency threshold: 0.50
Assess zero-delay cross-corr as the average of two delayed cross-corr (±τ)

Prefix of output file names: sample

Save intermediate results:
- [✓ Correlation diagrams]
- [✓ p-value diagrams]
- [✓ Efficiencies]
- Whole-sequences cross-corr. matrix

Enable parallel computing [✓]
Nr. threads: 4

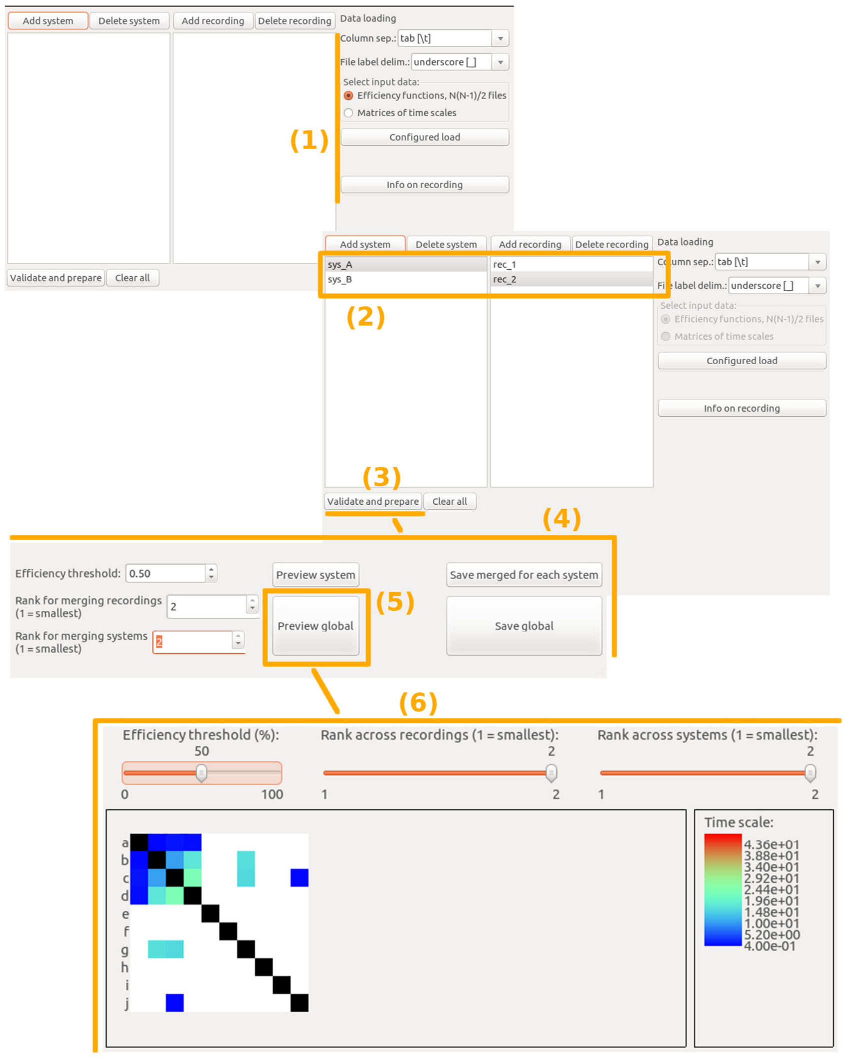
Significance threshold (1/1000): 50
Efficiency threshold (%): 50

Significance threshold (1/1000): 50
Efficiency threshold (%): 20

Fig. 5. Layout of the `netOnZeroDXC_analysis` app. The steps refer to example 1 and are highlighted with numbers: (1) input panel; (2) selection of the target output of the analysis; (3) parameters of the analysis; (4) settings concerning the output files; (5) analysis by means of parallel computing (enabled); (6) "preview" button to be used in order to show a preview of the matrix of time scales.

3.2. Merging of matrices

The steps of merging matrices from different recordings within the same system and, possibly, between different systems (see Fig. 3) is carried out by using the `netOnZeroDXC_merge` app. Within the app, the set of recordings/systems can be suitably handled. The app can load either matrices of time scales or efficiency functions. This latter possibility is provided in the case that the user carries out the previous analysis steps by means of the command line programs, which indeed yield efficiency functions. In the case of large datasets, and in order to avoid manual feeding of data, a "configured load" mode is available: a



Column sep.: tab [\t]
File label delim.: underscore [_]
Efficiency threshold: 0.50
Rank for merging recordings (1 = smallest) 2
Rank for merging systems (1 = smallest) 2
Efficiency threshold (%): 50 (0–100)
Rank across recordings (1 = smallest): 2 (1–2)
Rank across systems (1 = smallest): 2 (1–2)

Fig. 6. Layout of the `netOnZeroDXC_merge` app. The steps refer to example 3 and are highlighted with numbers: (1) input panel; (2) recordings/systems lists; (3) dataset validation; (4) panel with parameters settings, preview and output; (5) "global" preview button that leads to a merging both within system and between systems; (6) preview window with sliders to trim parameters.

suitable configuration file has to be provided to the app, in which recordings and systems are listed along with the filenames to load. Details on the configured load and a sample configuration file can be found in the user manual and in the examples folder, respectively.

4. Illustrative examples

In this section, four examples are presented to illustrate the functionalities of the package. These examples, as well as the corresponding data, can be found in the sub-folders of the "examples" folder provided with the package.

4.1. Analysis pipeline for a set of sequences

In the example contained in the folder "example_1", a complete analysis pipeline for a single recording is carried out. The analyzed dataset consists of 15 time series of 500 points each, stored as tab-separated columns inside a single file named "sequences.dat". The file is loaded in the app `netOnZeroDXC_analysis` by checking the "starting data" control box to "Sequences: single file". The parameters are set according to Fig. 5. The computation takes a few minutes on a standard laptop. Finally, a graphical preview of the computed matrix of time scales is shown up: since the analysis was carried out starting from raw multivariate time series, p-value diagrams are computed by the app as an intermediate step. Consequently, it is possible to trim both the significance threshold α and the efficiency threshold η_0. Fig. 5 shows a summary of the example, as well as the matrix of time scale for two different threshold configurations. Note that the p-value diagrams could have been computed also by using the command line program `netOnZeroDXC_diagram`. For each pair i, j, a call would look as follows:

```
netOnZeroDXC_diagram -i sequences.dat -o
sample_pdiag_i_j.dat-n <i> <j> -W 50 -L 4 -M 500
```

where `<i>` runs from 1 to 14 and `<j>` runs from `<i>` + 1 to 15.

4.2. Analysis pipeline for a set of p-value diagrams

In the example contained in the folder "example_2", an assessment of the matrix of time scale out of a set of p-value diagrams is carried out. This scenario occurs when p-value diagrams are already available. The diagrams are loaded in the app `netOnZeroDXC_analysis` by checking the "starting data" control box to "Diagrams of p-value: N(N-1)/2 files". The parameters are set according to the instructions (README file) contained in the example folder. Again, a graphical preview of the computed matrix of time scales is shown, as a function of the trimmable significance threshold α and efficiency threshold η_0.

4.3. Merging multiple recordings

In the example contained in the folder "example_3", a merging of multiple recordings and systems is carried out. The dataset consists of files containing efficiency function. Because $N = 10$ nodes are investigated, $N(N-1)/2 = 45$ files are present for each recording. The structure of the dataset is described in the configuration file "config_efficiency.cfg". Within this file, each row, which corresponds to a recording, contains the label of the system, the label of the recording, a mode label set to "e" (which stands for "efficiencies"), and a filename. The latter string identifies a file in the "data" folder that contains a list of efficiency files to be loaded for a given recording. The configuration file is loaded in the app `netOnZeroDXC_merge` by means of the "Configured load" button. The input controls are set according to Fig. 6. Once data are loaded, the dataset has to be validated and pre-processed. A graphical preview of the merged matrix is finally available, as a function of the trimmable ranking parameters that characterize the merging operations. Because efficiency functions are available, the efficiency threshold η_0 can be also trimmed. Fig. 6 shows a summary of the example and the merged matrix of time scales for a given choice of parameters. The folder of example 3 also contains a configuration file, "config_matrix.cfg", to load matrices instead of efficiency functions.

4.4. Merging matrices from multiple recordings

In the example contained in the folder "example_4", the merging of six recordings stemming from the same system is carried out. The dataset consists of six matrices of time scales, each containing 24 nodes. Again, the dataset is validated and pre-processed, and a graphical preview of the merged matrix is shown. In this case, because a single system is present, the only parameter that can be trimmed is the ranking parameter associated to recordings within a system.

5. Conclusions

In this work, the NetOnZeroDXC package is presented. The package implements a method for the identification of networks out of multivariate time series and is designed to be an easy-to-use interface between the user and the related multi-stage algorithm. The package provides two graphical interface apps to analyze multivariate time series data from a single source and to merge analysis from different sources, respectively. Two command line programs, each carrying out crucial computational steps implemented in the first, analysis app, are also included in the package. As an alternative to the analysis app, these command line programs are suitable for analysis run on computer clusters.

The package can be applied in all research fields where each node of a complex systems is described by a recorded time series and the existence, as well as the timing features of an underlying network structure is investigated.

A future development of the method presented in this paper can regard the replacement of the cross-correlation with mutual information in order to investigate nonlinear correlations. Alternatively, partial correlations can be investigated in order to possibly distinguish direct links from mediated ones.

A major improvement of the method would then concern the evaluation of nonzero delay cross-correlations in order to assess not only network structures but also hierarchical relations within a network.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Buckner RL, Andrews-Hanna JR, Schacter DL. The brain's default network: anatomy, function, and relevance to disease. Ann New York Acad Sci 2008;1124(1):1–38. http://dx.doi.org/10.1196/annals.1440.011.

[2] Tsonis AA, Swanson KL, Roebber PJ. What do networks have to do with climate? Bull Am Meteorol Soc 2006;87(5):585–95. http://dx.doi.org/10.1175/BAMS-87-5-585.

[3] Gelbrecht M, Boers N, Kurths J. A complex network representation of wind flows. Chaos 2017;27(3):035808. http://dx.doi.org/10.1063/1.4977699.

[4] Mantegna RN. Hierarchical structure in financial markets. Eur Phys J B 1999;11(1):193–7. http://dx.doi.org/10.1007/s100510050929.

[5] Perinelli A, Chiari DE, Ricci L. Correlation in brain networks at different time scale resolution. Chaos 2018;28(6):063127. http://dx.doi.org/10.1063/1.5025242.

[6] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C, 3rd ed.. Cambridge University Press; ISBN: 978-0-521-88068-8, 2007.

[7] Franchi M, Ricci L. Statistical properties of the maximum Lyapunov exponent calculated via the divergence rate method. Phys Rev E 2014;90(6):062920. http://dx.doi.org/10.1103/PhysRevE.90.062920.

[8] Perinelli A, Ricci L. Identification of suitable embedding dimensions and lags for time series generated by chaotic, finite-dimensional systems. Phys Rev E 2018;98(5):052226. http://dx.doi.org/10.1103/PhysRevE.98.052226.

[9] Brookes MJ, Woolrich MW, Price D. In: Supek S, Aine C, editors. Magnetoencephalography. Berlin: Springer-Verlag; 2014, p. 321–58, Ch. An Introduction to MEG Connectivity Measurements, http://dx.doi.org/10.1007/978-3-642-33045-2_16.

[10] Schreiber T, Schmitz A. Improved surrogate data for nonlinearity tests. Phys Rev Lett 1996;77(4):635–8. http://dx.doi.org/10.1103/PhysRevLett.77.635.

[11] Schreiber T, Schmitz A. Surrogate time series. Physica D 2000;142(3–4):346–82. http://dx.doi.org/10.1016/S0167-2789(00)00043-9.

[12] Räth C, Gliozzi M, Papadakis IE, Brinkmann W. Revisiting algorithms for generating surrogate time series. Phys Rev Lett 2012;109(14):144101. http://dx.doi.org/10.1103/PhysRevLett.109.144101.

[13] Galassi M, et al. GNU Scientific Library Reference Manual, 3rd ed.. ISBN: ISBN 0954612078, 2009.

[14] GSL website: https://www.gnu.org/software/gsl/.

[15] wxWidgets library website: https://www.wxwidgets.org/.